Enterprise Identifiers for Global Naming Across the C4I-Simulation Boundary

Sam Chamberlain
U.S. Army Research Laboratory
ATTN: AMSRL-CI-C
Aberdeen Proving Ground, MD 21005-5066
410-278-8948

wildman@arl.army.mil
http://www.arl.army.mil/~wildman

Keywords:
Enterprise Identifiers, Default Operational Organizations

ABSTRACT: *In any information system, a critical feature is the ability to link together disparate pieces of data and information via relationships. One way to greatly facilitate this task is to provide a common technique for identifying the pieces so that they can be conveniently referenced. This can be accomplished by standardizing one field across disparate data sources. Perhaps no simpler enhancement can produce such a huge benefit with as little intrusion into legacy systems. This is the objective of enterprise identifiers. If data can be globally identified using a common scheme, then one can spontaneously "plug and play" disparate, arbitrary pieces of information without prior coordination.*

1. Model-Based Battle Command

Model-based battle command [1] refers to the notion that battle command information must be maintained in a form conducive to machine manipulation if one is to truly automate its processes. In other words, one must model the battlefield to achieve true automation [2]. This implies that the tasks of *doing* battle command and that of *simulating* battle command should ultimately be united:

> Battle command system developers have traditionally included modeling and simulation (M&S) as a major element of their acquisition strategy (in fact, it is now mandated). But it is interesting and bothersome that battle command is viewed so differently between the system development and M&S communities. A major disparity exists between the way we *do* battle command and the way we *model* battle command. Although numerous research programs have produced a diverse set of battle command models and simulations, they are still relegated to a role of support and study rather than being considered as part of the actual system. Under a model-based paradigm of battle command, the system development and M&S communities unite so that using automated tools to do battle command is merely simulating battle command with real-time, real-world data. Simply stated, from an automation perspective: modeling battle command should differ little from actually "doing" battle command.[1]

In a model-based battle command scheme, the model, and its container, the database[2], become the center (rather than just a supporting resource) of the design and implementation of the automated portion of the battle command system. Any element of a force, down to a single warrior, may maintain its own copy of the battlefield model in its local computing environment. This is now possible because of the amazing computational power and storage capacity available in small packages.

This allows the database to serve as the conduit, or hub, by which information is transferred between different units (i.e., nodes) and often, between the applications within the same unit; see Figure 1. Currently, abstractions are manifested by a variety of "information models" (also called data models) that may be defined using formal

[1] From [1], page 31.

[2] The term "database" is used as a general concept and does not imply a single, monolithic entity.

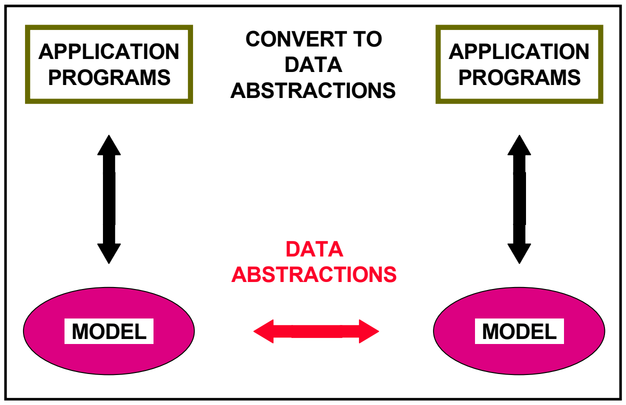


Figure 1: The Battlefield Model is the Core of the Automation Process

representations conducive to database implementations. Common examples are IDEF1X [3] for relational models and UML for object-oriented models [4].[3]

The information in the model is stored in an abstract form conducive to machine manipulation rather than in a form suitable for direct human assimilation. However, human-oriented information, such as text, images, digitized audio, or graphic icons can be easily attached to the abstractions. But it is the abstractions that form the backbone of the battle command information infrastructure. Users rarely (if ever) directly access the data; instead, they rely on sophisticated, specifically tailored application programs, that in turn, access the database (or model). Thus, the user is completely unaware that an abstract data model exists or that transactions have occurred. From the data structures, it should not be evident whether the information is real or simulated.

Among information systems, whether a C4ISR system or a simulation, a critical feature is the ability to link together disparate pieces of data and information via relationships. One way to greatly facilitate this task is to provide *a common technique for identifying the pieces so that they can be conveniently referenced.* Standardizing one field across disparate databases, and then including that field in all data objects or entities can accomplish this. Perhaps no simpler enhancement can produce such a huge benefit with as little intrusion into legacy systems. This is the objective of enterprise identifiers. If data can be globally identified using a common scheme, then one can spontaneously "plug and play" disparate, arbitrary pieces of information without prior coordination.

[3] *IDEF1X*: **I**CAM (Integrated Computer Aided Manufacturing) **Def**inition **1** – E**x**tended.
UML: **U**nified **M**odeling **L**anguage

2. Enterprise Identifiers (EID).

An identifier is a property that identifies something. Different techniques may be used to accomplish the task of identification, but regardless of the technology, there must be some way to identify and reference data stored in computers. Databases are commonly used to store data and they may be based on one of several models. The two most common models are the relational and object-oriented models that identify items using *primary keys* (PK) and *object identifiers* (OID), respectively. A major accomplishment towards interoperability would be the acceptance of a common identification scheme that spans both of these data storage technologies.

In the relational model, data is stored in tables with attributes (i.e., columns) and the rows of a table may represents an entity or a relationship between entities. Every row of a table must be uniquely identifiable. A *candidate key* is a set of attributes that accomplishes this task. There may be several candidate keys for a table, and one of these is selected as the PK for the table. A property of this approach is that a PK may be composed of several attributes. Further, these attributes may be imported from other tables, they may contain codes or symbols that provide insight into the properties of the item they identify, and the PK may be different for every table. Although these characteristics may be viewed as "features" by some designers, they introduce significant inertia and complexity when one attempts to modify or share PKs across disparate databases. This is referred to as the "foreign key ripple effect" in [5].

To alleviate some of these problems, the concept of surrogate keys was introduced [6],[7],[8]. A *surrogate key* (SK) is a PK that is composed of a single attribute with no intelligence built into it. This means that it can not be imported from another table and that nothing can be inferred from the SK about the item it identifies. However, a SK only needs to be unique within a single table. The same SK may exist in numerous tables (e.g., a common approach is to number rows from 1 to N in every table).

Enterprise keys expand upon this concept. An enterprise key (EK) is a surrogate key that is unique across all tables of the enterprise [9]. Therefore, if the enterprise were the Army, then every row of every table of every database would be uniquely identified via an EK. This is a powerful property when the challenging task of integration is pursued.

An *enterprise identifier* (EID) is the generic version of an EK. A goal is to develop a scheme that is independent of any database technology. An EID is special because it uniquely identifies an entity or object across the entire enterprise, not just within a specific domain or type of database. Similarly, if two data items have the same EID, then they must be semantically equivalent, that is, they must represent the same thing. This allows different formalisms to be used to represent identical items. It does not matter if the item is represented as a row in a relational database (with an EK) or as an object (with an Object ID); if two items have the same EID, then they are semantic equivalents.

The two primary challenges to accomplish an EID implementation are (1) the development of a flexible EID allocation scheme that can be as distributed or centralized as desired, and (2) reaching agreement on a common structure usable by any database technology. These issues will be addressed in a later section.

3. Default Operational Organizations

3.1 Stationary Data

Data may be categorized based upon its perishability. *Reference data* is static data. Examples are lookup table entries for state, airport, or country codes. At the other extreme is *dynamic data*. As its name implies, it is constantly changing and must be updated frequently to remain synchronized. Situational awareness (SA) data is in this category. The third category lies between these two extremes. *Stationary data* is information that is "semi-static," meaning that it is relatively invariant over its lifetime so that it can be reasonably maintained in a shared reference library. An example of stationary data is phone numbers. Although thousands of phone numbers are added, deleted, and changed daily worldwide, it is not a frequent event for a person's phone number to change. Usually, a person's phone number is static for the duration of their tour of duty. Therefore, phone numbers can be reasonably maintained in a reference library called a phonebook that is only distributed once per year. The small subset of new, changed, and deleted phone numbers are handled via other means (i.e., directory assistance).

Several C4ISR[4] oriented data models (DM), related to the C2 Core DM[5], include battlefield entities in five basic domains: organization, materiel, personnel, facility, and features. These five domains contain large amounts of information that can be considered stationary. This means that the data can reside in common reference libraries (e.g., provided via servers) that can be periodically downloaded into one's computers; see Figure 2. By rigorously controlling the update process of the stationary data, users can be confident that they have a consistent set of reference material preloaded into their computers. If EIDs are used to provide a common naming scheme across the reference libraries (i.e., for both the reference and stationary data), then the common data will include a common set of EIDs. This allows users to exchange the terse EIDs instead of the bulky data and can significantly reduce bandwidth requirements for passing C4ISR information.

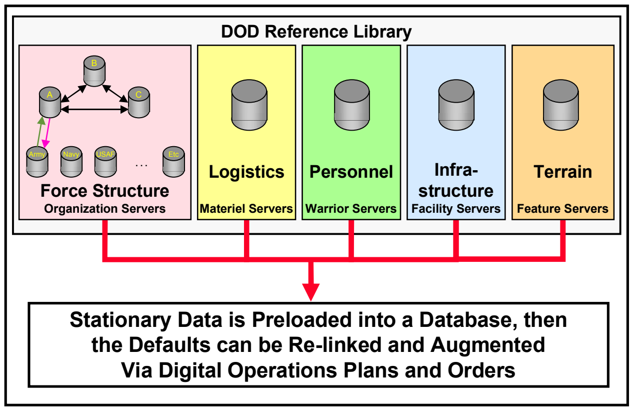


Figure 2: Suite of Servers that Contain Stationary Data

However, for reference and stationary data to be consistent, a proponent for the data must be identified to rigorously control the creation, maintenance, and deletion of the data. A reasonable practice is to select the agencies that already maintain the data as part of their charter. For example, every service has a force structure development community that handles a wide variety of force structure documents. These agencies should be the owners of the stationary and reference data for organizations. The Defense Logistics Agency (DLA) already maintains the FEDLOG catalog via its Defense Logistics Information Service (DLIS)[6]. This should be the authoritative source for materiel related data that has national stock numbers. Although the lines of ownership are rarely perfectly clear for stationary data, it is important that authoritative sources be identified, sanctioned, and funded to provide a consistent product.

[4] C4ISR: Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance.

[5] Command & Control (C2) Core Data Model: http://www-datadmn.itsi.disa.mil/ddm.html

[6] See http://www.dlis.dla.mil/govord.htm for information.

3.2 Organization Data as Tree Graphs

The concept of *default operational organizations* (DOO) is introduced in [10] and is based upon three fundamental tenets. First, the concept of *force structure* lies at the heart of the representation of battle command. Second, a default force structure exists that is composed of a set of default organizations that are linked together with a default command structure (DCS). This force structure is relatively stable (i.e., it is stationary data) and if designed properly (i.e., is richly populated), then it can be used as the base structure for integrating entities and building arbitrary orders of battle. Third, operational command structures (i.e., unit task organizations) are fluid, and are nearly always constructed by modifying (i.e., re-linking the nodes of) the default force structure.

Force structure can be formally represented using tree graphs, often used for organization charts (org chart); see Figure 3.[7] The terms *organization*, *command structure*, and *unit* are mapped to the formal definitions for node, a set of links, and tree, respectively. In. other words, a unit (U) is composed of a set of organizations (O) and a set of links called a command structure (CS).[8] The purpose of the set of links in the DCS is to define aggregation (or composition). One can interpret the set of links using the phrase "is composed of," e.g., "a 'is composed of' b, c, and d." Set E in Figure 3 could represent a DCS.

The concept of DOOs states that default org charts for military units do indeed exist, but to be truly useful, they must reflect the way a unit deploys or fights. In other words, the structure must include all the nodes typically used when the unit is deployed. Currently, force structure documents are administrative in nature. In the Army, these are called Tables of Organization and Equipment (TOE), or Modified TOEs (MTOE). It is proposed that the information in these documents be enhanced by imposing a formal, hierarchical structure to it (i.e., making it into a tree graph), beginning from some arbitrary root node, say the Department of the Army, and extending down to the individual warrior level, commonly called a billet. However, to be truly useful, the structure must include the nodes that are used on a daily basis for operations, such as teams, sections, elements, and squads. Currently, these organizations are normally defined in Field Manuals that describe tactics, techniques and procedures. Therefore, an objective is to evolve the force structure documents from an administrative purpose to one that is operational in nature.

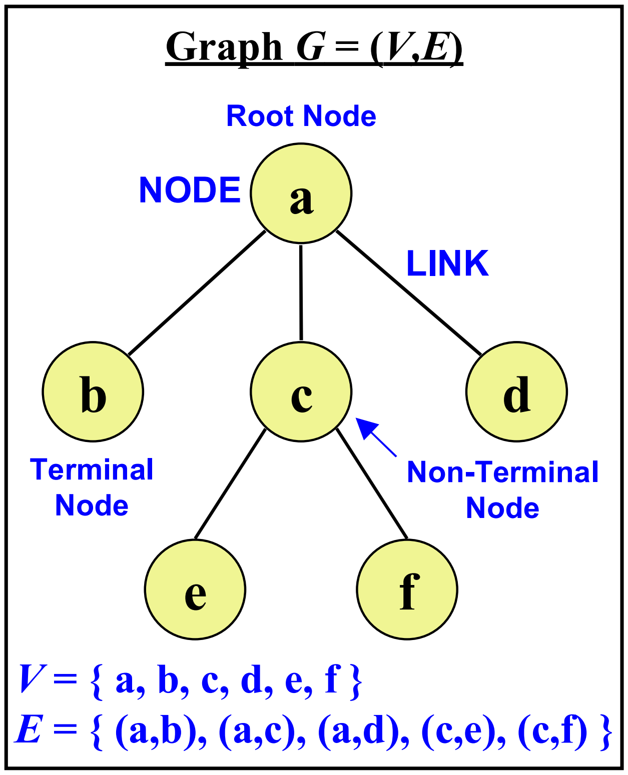


Figure 3: A Tree Graph with 5 Nodes

To transition to a formal, consistent, hierarchical representation of force structure, semantics must be developed. Some recommended semantics follow.

(A) A node may be a member of several command structures simultaneously.

(B) Internal nodes are aggregation points, and therefore, should have multiple children. There are two type of internal nodes that correspond to two motives for aggregation:

  i. *Crew Organizations* that provide a 1:1 correspondence with a piece of equipment requiring operation by one or more billets.

  ii. *Doctrinal Organizations* that provide an arbitrary aggregation point due to doctrine, tactics, techniques, or procedures

(C) Personnel positions ("billets") are nodes just like other organizations. They are always leaf nodes (i.e., they can not be internal nodes), but the reverse is not strictly so. (Note: people are *assigned to* billets, but are not billets.)

---

[7] Recall that a graph $G(V,E)$ is composed of a set of vertices (or nodes) v and a set of edges (or links) e. A *tree* is a connected graph with no cycles, meaning that all its nodes are connected to at least one other node, and only one path exists between any two nodes.

[8] $U(O,CS)$ is mapped to $G(V,E)$.

(D) Internal nodes can be active (in use) or dormant, but an active internal node must have a designated leader (i.e., a reference to a billet).

The result of applying these and other rules is the development of a consistent force structure tree whose links describe the aggregation (or composition) of three types of nodes: doctrinal organizations, crews, and billets. The default command structure presents the organic, or "at rest" perspective of the tree. However, numerous other links may be added to provide more sophisticated relationships between organizations and other entities. The options are limitless.

If the doctrinal and crew organizations are correctly and fully populated, then (ideally), any order of battle may be configured by simply rearranging the existing set of organizations without creating any new ones. New *links* may have to be created, but no new *nodes* (i.e., organizations). The set of organizations that meet this requirement is called the set of DOOs, or just DOOs; Set V in Figure 3 could be a DOO. To reiterate, the objective is to produce a set of organizations (i.e., nodes), from which numerous command structures can be applied to meet the requirements of any mission without having to create more aggregation points. One command structure is selected as the default, but this in no way limits the number of command structures that can be created to meet mission needs.

4. Organization Identifiers

Given that a default force structure exists, it is now simple to enumerate the DOOs (or nodes of the tree). When a new organization is created by the force development community, it is assigned an identifier that stays with the organization for its life (i.e., until is it disestablished). This identifier is called an *organization identifier*, or simply *Org-ID*. This allows the default force structure (tree) to be represented via a set of organizations, each with an Org-ID (stored in an organization table), and a set of default links (stored in an organization-association table) that connects every organization with a default parent, except, of course, for the root node that has no parent. This structure can be maintained in a database, called the organization server (or org server) which is populated and maintained by the force development community. This server becomes the authoritative source of current force structure data for the organizations that it includes. Org-IDs are EIDs, and therefore, are unique across the whole enterprise. Anyone who downloads all or part of the force structure tree will receive the same set of DOOs with their DCS.

A debatable question is how large should an Org-ID be? The objective is to make it big enough so that it doesn't repeat for a long time, yet keep it small enough to maintain performance and minimize bandwidth overheads. The current choice is 32 bits. This allows 4.3 billion organizations to be labeled provided that they are not wasted. This is enough for over 200,000 Army divisions[9].

A two-tiered approach for allocating Org-IDs has been implemented that is composed of organization servers that maintain the force structure data, and Org-ID servers that allocate Org-IDs; see Figure 4. When a force structure

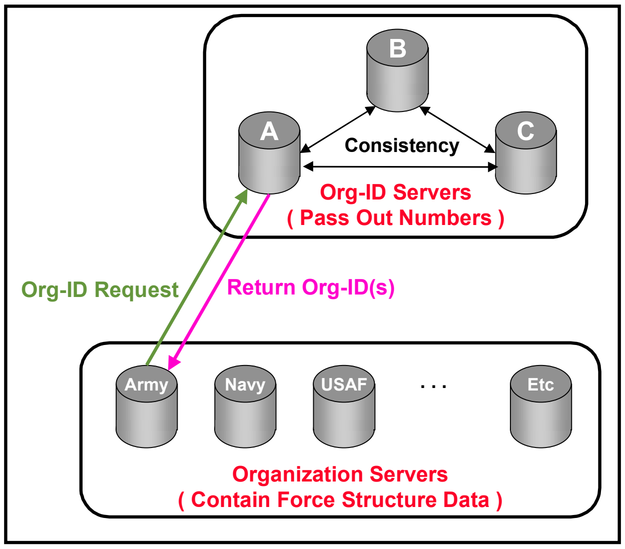


Figure 4: Two Tier Architecture for Servers

developer creates a new organization in an org server, the org server must request one or more Org-IDs from an Org-ID server. Org-ID servers pass out Org-IDs on a first-come, first-served basis to prevent waste. There are no reasons to pre-allocate blocks of Org-IDs because they are simply identifiers.[10] An important feature is that Org-ID servers do not contain any force structure data. All that is maintained in an Org-ID server is the identity of the org server to which an Org-ID was given, and whether the Org-ID has been activated. This design allows any organization, service, or country to participate in the process without having to share its force structure data. The owner of an org server maintains complete sovereignty over its force structure data.

[9] For an analysis, see [10], Appendix B.

[10] In contrast to IP addresses or phone numbers that infer hierarchical routing information.

It must be emphasized that organization template data, such as TOEs, do not receive Org-IDs. Technically, this information resides in the *organization-type* domain and is tagged with *organization-type-ids*. The details of these tags will be covered in the discussion of ad hoc organizations.

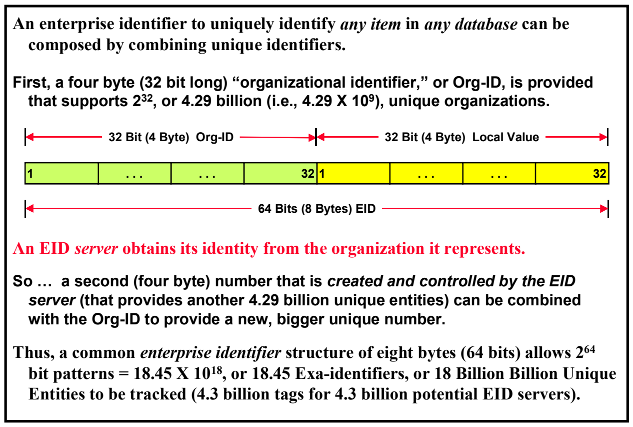
5. An EID Server Architecture

As previously mentioned, a primary challenge in creating an EID allocation system is developing an approach to guarantee that EIDs are unique while ensuring that they can be easily obtained. Whenever a database management system creates a piece of data (e.g., inserts a row into a table), it must obtain a globally unique EID to tag the data. This must be accomplished rapidly because significant delay is unacceptable. Further, all EIDs must have the same structure and be sized to be large enough to handle the vast number of required cases without being excessively large to cause performance degradation or a communications burden on C4ISR systems.

Note that Org-IDs are EIDs, and therefore, they can be used to implement a more expansive EID scheme across an enterprise. Concatenating a second unique value to an Org-ID also produces a value that is unique across the enterprise, that is, another EID. Consider a technique where any official organization (i.e., one with an Org-ID) is allowed to produce larger EIDs by concatenating a locally selected value to its Org-ID. If the organization is careful never to repeat the locally selected value, the concatenated values will also be EIDs. Therefore, a standard EID structure, to be used for all data identification purposes, can be defined as a 64 bit identifier composed of a 32 bit Org-ID followed by another 32 bit integer selected and controlled locally. In other words, a common, 64-bit EID data structure can be defined for use by any database for any type of data. This would allow 18 billion billion different identifiers to be created; see Figure 5.

A computer program that provides EIDs is called an EID server. Any official organization may operate an EID server and control who may use it. It doesn't matter whether the EID is used for an actual command and control system or a simulation. The result is that any piece of data can be uniquely identified and referenced across arbitrary systems without previous coordination.

The elegance of this approach is its flexibility. Consider some extreme cases. Theoretically, there could be one big EID server run by the Department of Defense. Every organization in the enterprise would be forced to obtain EIDs from this server; see Figure 6. At the other extreme, there could be 4.3 billion EID servers, one for each possible organization and it could be limited to serve only one organization; see Figure 7. Of course, the real answer will be in the middle of these extremes. Each organization controls its EID servers based upon local policy decisions.


An enterprise identifier to uniquely identify *any item* in *any database* can be composed by combining unique identifiers.

First, a four byte (32 bit long) "organizational identifier," or Org-ID, is provided that supports 2^{32}, or 4.29 billion (i.e., 4.29 X 10^9), unique organizations.

An EID *server* obtains its identity from the organization it represents.

So ... a second (four byte) number that is *created and controlled by the EID server* (that provides another 4.29 billion unique entities) can be combined with the Org-ID to provide a new, bigger unique number.

Thus, a common *enterprise identifier* structure of eight bytes (64 bits) allows 2^{64} bit patterns = 18.45 X 10^{18}, or 18.45 Exa-identifiers, or 18 Billion Billion Unique Entities to be tracked (4.3 billion tags for 4.3 billion potential EID servers).

Figure 5: Org-ID Based Expansive EID

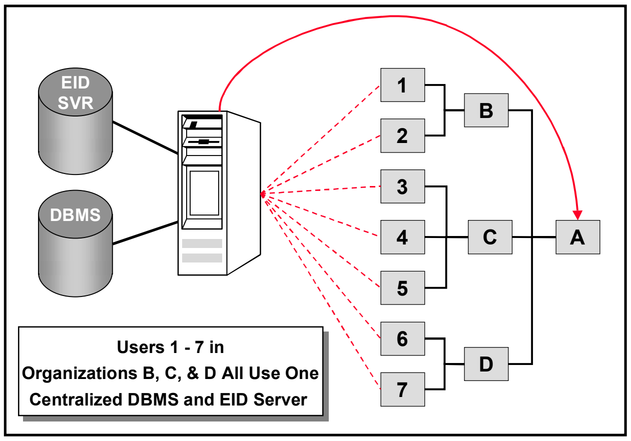

EID SVR
DBMS
1
2
3
4
5
6
7
B
C
D
A
Users 1 - 7 in Organizations B, C, & D All Use One Centralized DBMS and EID Server

Figure 6: Extreme One - Centralize Control

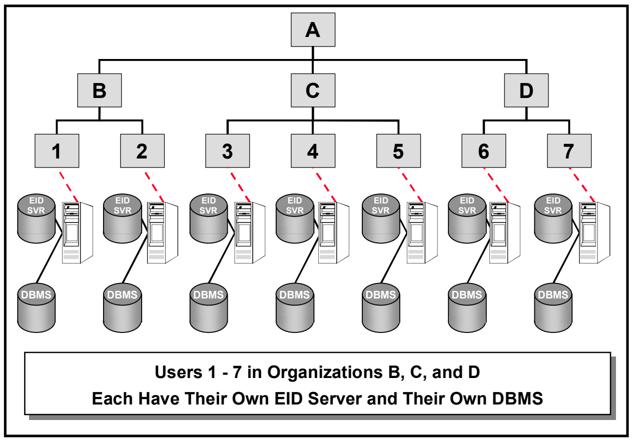

A
B
C
D
1
2
3
4
5
6
7
EID SVR
DBMS
Users 1 - 7 in Organizations B, C, and D Each Have Their Own EID Server and Their Own DBMS

Figure 7: Extreme Two - Decentralized Control

When an official organization decides to establish an EID server using its Org-ID, it also determines who will be allowed to access it. In the simplest case, it can provide a *single user* EID service. This would be the preferred approach for isolated systems, such as those with individual warriors (e.g., forward observers or special operations teams). *Single user* EID servers can be very simple and simply allocate increasing values. *Multi-user* EID servers can be as complex as desired. They can track to whom EIDs are allocated just like an Org-ID server. They can also require the requesting database to provide detailed information before they allocate the EID. For example, data insertions require an EID so this technique could be used to control the creation of data across a wide variety of distributed systems. The extent of centralized or decentralized control is completely flexible and can be adjusted at any time; see Figure 8.

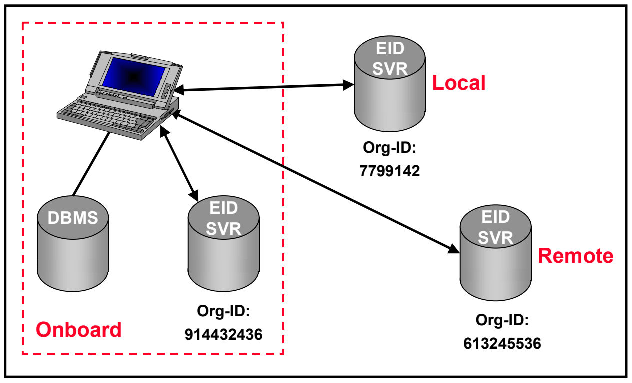


Figure 8: EID Server Access is Determined by Owner

This is consistent with the EID definition. Under this proposal, all data is identified using a 64 bit EID. However, recall that Org-IDs are only 32 bits long. It is awkward to have organizations tagged with different sized EIDs from every other type of data. Therefore, when entered into a database, Org-ID would be cast into standard sized, 64 bit *Org-EIDs* by simply concatenating a predefined 32 bit value to them[11]. This makes all EIDs the same structure. EIDs for ad hoc organization are not created this way. Like any other piece of data, they must obtain an EID from an EID server before they can be inserted into a database[12]. Consequently, official organizations can be identified by their unique EID suffix, although this is strongly discouraged. However, it should be clear that an ad hoc organization requires all 64 bits to be identified, and therefore, can not be used as a prefix for 64 bit EIDs. This prevents data from being inserted under the sponsorship of ad hoc organizations; see.Figure 9.

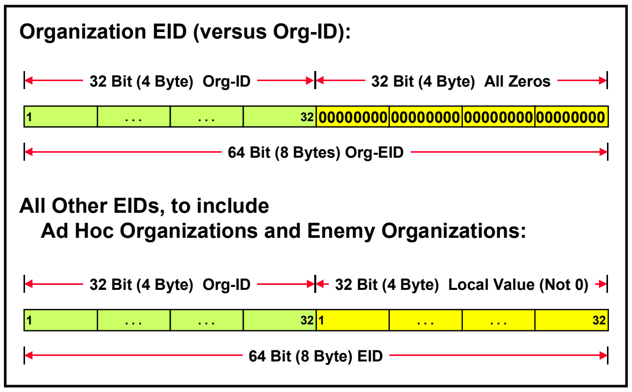

Organization EID (versus Org-ID):
32 Bit (4 Byte) Org-ID
32 Bit (4 Byte) All Zeros
1
...
...
32
00000000
00000000
00000000
00000000
64 Bit (8 Bytes) Org-EID
All Other EIDs, to include
Ad Hoc Organizations and Enemy Organizations:
32 Bit (4 Byte) Org-ID
32 Bit (4 Byte) Local Value (Not 0)
1
...
...
32
1
...
...
32
64 Bit (8 Byte) EID

Figure 9: Org-EIDs Versus Org-IDs

6. Ad Hoc Organizations

So far, only official organizations that are created by bona fide force developers with Org-ID assignment authorization have been discussed. However, when required, anyone may be allowed to create unofficial, or ad hoc, organizations. Of course, dissemination of ad hoc organization information is completely the responsibility of the creating organization.

Ad hoc organizations are completely interchangeable with official organizations. They can be associated with any other entity or object that an official organization may be associated. One can track ad hoc and official organizations using all the same applications. The only restriction on ad hoc organizations is that they may not be used to establish EID servers. In other words, one can not insert new data under the auspices of an ad hoc organization. Only official organizations can sponsor the creation of data (via their EID servers).

Enemy organizations are ad hoc organizations. This is because the enemy has decided not to participate in the Org-ID process. Therefore, enemy organizations are perceptions and are entered using EIDs obtained from the EID servers of those who are authorized to enter enemy order of battle data (i.e., the intelligence community).

Recall that organization-types (org-types) are the templates from which organizations (orgs) are instantiated. Org-types are used to represent TOE/MTOE data. Org-types get EIDs just like any other non-organization entity or object. They receive their EIDs

11 An obvious choice is 32 zeros. For the Joint Common Database, the value 1 was chosen.

12 This means that an EID server can not use the value selected as the predefined 32 bit suffix. This reducing the number of available EIDs by one (i.e., to 4.3 billion – 1) for each EID server.

from the EID servers of the organizations authorized to create force structure data (i.e., from the force development community). From an EID perspective, it is important to understand the distinction between official organizations, ad hoc organizations, and org-types. Only official organizations receive Org-IDs (32 bit) from the Org-ID servers, the others receive standard EIDs (64 bits that are created from Org-IDs) just like any other entity.

Anyone may create ad hoc organizations, and the simulation community is expected to be a primary exploiter of this feature. Technically, it is perfectly permissible to make a copy of a friendly or enemy organization-type or organization hierarchy and apply one's own EIDs to differentiate it from the original data. Once the EIDs are changed, the data becomes independent from the original data and is under the control of the organization that created it (i.e., applied the new EIDs). The new organizations are no longer official. They become ad hoc organizations and can be modified to meet the needs of their creator, to include deleting or adding other ad hoc organizations. Clearly, this principle can be easily applied to all entities. This is exactly what the simulation community requires: access to stationary data that can be easily modified to meet its own needs without infringing on the sovereignty of the original data owners.

7. Summary

This paper presented four points. First, whether in a C4ISR system or a simulation, one of the most fundamental tasks is the unique identification of data items and components. Providing a common identification method is imperative for maintaining integrity within and across both C4ISR systems and simulations. Second, enterprise identifiers (EID) are one approach to providing unique values across disparate systems. EIDs have a common structure and are unique across the enterprise. Third, a primary requirement for a successful EID allocation system is that it is flexible, and when required, completely distributed; it can not produce bottlenecks nor can it be the source of delay. Fourth, to accomplish these characteristics, a suite of EID servers can be implemented that are based upon the concept of default operational organizations (DOO). DOOs are the nodes of a default force structure hierarchy maintained by the force development community. Each DOO is assigned an organization identifier (Org-ID) that is also unique across the enterprise and exists for the lifetime of the organization. Finally, there are large amounts of information that are stable enough to be maintained in reference libraries. This information may be static (reference data) or semi-static (stationary data). Reference material can be assigned common EIDs, but to maintain consistency, proponents for the information must be identified to control its content.

These characteristics are independent of whether the information is used for real world or simulation purposes. This is transparent to the users. The common theme is that all the pieces of information are uniquely identified using a common scheme allowing one to find, reference, and retrieve information regardless of the rest of its format. The remaining format information can be easily obtained with syntactic languages, such as XML[13]. Together, these standards allow one to spontaneously "plug and play" disparate, arbitrary pieces of information without prior coordination.

[13] XML: The Extensible Mark-up Language, a standard for describing the structure of information; see http://www.w3.org/XML/.

8. References

[1] S. Chamberlain: "Model-Based Battle Command: A Paradigm Whose Time Has Come" *Proceedings of the First International Symposium on Command and Control Research and Technology*, pp 31-38, National Defense University; 19-22 Jun 1995. http://www.arl.army.mil/~wildman/PAPERS/tr2172.html

[2] M. Boller, "Common Understanding for Transformation Brigades," *Military Review*, Sep-Oct 2000. http://www-cgsc.army.mil/milrev/English/SepOct00/boller.htm

[3] T. A. Bruce. "Designing Quality Databases with IDEF1X Information Models." Dorset House Publishing, New York, NY, 1992.

[4] G. Booch, I. Jacobson, J. Raubaugh. OMG Unified Modeling Language Specification (Version 1.3, March 2000). http://www.omg.org/technology/documents/formal/omg_modeling_specifications_avai.htm.

[5] T. Johnston: "Primary Key Reengineering Projects: The Problem;" *DM Review*, February, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=1866.

[6] E.F. Codd: "Extending the Relational Model to Capture More Meaning,'" *ACM Transactions on Database Systems, Vol 4(4)*, Dec 1979.

[7] C.J. Date: "Relational Databases: Selected Writings," Addison-Wesley, Reading, MA, 1986.

[8] M. Lonigro, Mike: "The Case for the Surrogate Key", *Intelligent Enterprise Database Programming and Design On-line,* May 1998 http//:www.dbpd.com/vault/9805xtra.htm.

[9] T. Johnston: "Primary Key Reengineering Projects: The Solution;" *DM Review*, March, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=2004

[10] S. Chamberlain: "Default Operational Representations of Military Organizations," Army Research Laboratory Technical Report: ARL-TR-2172; February 2000. http://www.arl.army.mil/~wildman/PAPERS/tr2172.html

Author Biographies

SAM CHAMBERLAIN is a computer scientist at the U.S. Army Research Laboratory, Aberdeen Proving Ground, Maryland where he has worked for the past 19 years. He has a Ph.D. in computer science from the University of Delaware and a Ranger Tab from the U.S. Army.